AD_____

REPORT NO 5/88

USING IMSL MATHEMATICAL AND STATISTICAL COMPUTER SUBROUTINES IN PHYSIOLOGICAL AND BIOMECHANICAL RESEARCH

U S ARMY RESEARCH INSTITUTE
OF
ENVIRONMENTAL MEDICINE
Natick, Massachusetts

OCTOBER 1987

UNITED STATES ARMY
MEDICAL RESEARCH & DEVELOPMENT COMMAND

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

DISPOSITION INSTRUCTIONS

Destroy this report when no longer needed.

Do not return to the originator.

Report No. T5/88

Using IMSL mathematical and statistical computer subroutines in physiological and biomechanical research

Everett Harman
US Army Research Institute of Environmental Medicine
Natick, MA 01760-5007

October 1987

SECURITY CLASSIFICATION OF THIS PAGE

AD-A192 394

REPORT DOCUMENTATION PAGE

Form Approved OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION: US Army Research Institute of Environmental Medicine	6b OFFICE SYMBOL (If applicable): SGRD-UE-PH
7a NAME OF MONITORING ORGANIZATION: Same as 6a.	
6c ADDRESS (City, State, and ZIP Code): Natick, MA 01760-5007	7b ADDRESS (City, State, and ZIP Code)
8a NAME OF FUNDING / SPONSORING ORGANIZATION	8b OFFICE SYMBOL (If applicable)
9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO
6.2	3E162777A879	879B	125

11 TITLE (Include Security Classification)			
12 PERSONAL AUTHOR(S)			
13a TYPE OF REPORT: Technical Report	13b TIME COVERED: FROM May 1987 TO Sep 87	14 DATE OF REPORT (Year, Month, Day): October 1987	15 PAGE COUNT: 55
16 SUPPLEMENTARY NOTATION			

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Computer software, physiology, biomechanics

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

IMSL Corporation sells a widely used and respected package of mathematical and statistical computer subroutines written in the FORTRAN language. While engineers, mathematicians and others who work regularly with computation are familiar with the usefulness of the subroutines most medical researchers are neither aware of how the package could be useful to them nor knowledgeable enough in FORTRAN to be able to write programs to call the routines. This report gives examples which illustrate the applicability of IMSL subroutines to research in physiology and biomechanics, and shows how to write simple FORTRAN programs to define variables, read data from a file, call the IMSL subroutines, and store or display results. Increased usage of IMSL software by medical researchers can lead to more sophisticated and quantitative treatment of data, and improve the overall quality of research.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION
22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code) / 22c OFFICE SYMBOL

DD Form 1473, JUN 86 — Previous editions are obsolete — SECURITY CLASSIFICATION OF THIS PAGE

ACKNOWLEDGEMENTS

Gratitude is expressed to personnel at IMSL Incorporated who provided information about their company's software products and granted permission for excerpts of their documentation to be reprinted in this report.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

IMSL Corporation sells a widely used and respected package of mathematical and statistical computer subroutines written in the FORTRAN language. While engineers, mathematicians and others who work regularly with computation are familiar with the usefulness of the subroutines, most medical researchers are neither aware of how the package could be useful to them nor knowledgeable enough in FORTRAN to be able to write programs to call the routines. This report gives examples which illustrate the applicability of IMSL subroutines to research in physiology and biomechanics, and shows how to write simple FORTRAN programs to define variables, read data from a file, call the IMSL subroutines, and store or display results. Increased usage of IMSL software by medical researchers can lead to more sophisticated and quantitative treatment of data, and improve the overall quality of research.

INTRODUCTION

IMSL Incorporated sells a computer software product called the IMSL Library which is a comprehensive collection of over 600 mathematical and statistical FORTRAN subroutines. Since its creation in 1970 IMSL has become internationally recognized by government, industry and academia as a comprehensive, reliable resource in the field of numerical computing.

There are three sub-libraries within the IMSL software package:

MATH/LIBRARY - general applied mathematics

STAT/LIBRARY - statistics

SFUN/LIBRARY - special functions

To use any of the routines, a program must be written in FORTRAN to define variables, read data from a computer file, call the IMSL subroutines, and store or display results. The purpose of this report is to show how simple FORTRAN programs can be written to access the IMSL library. Illustrative examples are provided of applications to research in physiology and biomechanics.

An alternative product to the IMSL library, called PROTRAN, produced by the same company, may be of interest to some users. It allows those with no programming knowledge to solve mathematical and statistical problems. The system is basically a program which converts a set of relatively simple commands provided by the user into FORTRAN computer code which calls the appropriate IMSL subroutines. While the system may prove useful in an environment where users are unfamiliar with programming, there are some disadvantages to PROTRAN over the regular IMSL library. PROTRAN requires

considerable computer memory and computing time to pre-process the simplified commands, and is therefore limited to larger computers. Unlike the standard library, it will not operate on an IBM PC compatible machine, and IMSL has no plans to develop a PROTRAN version for the PC in the near future. The standard library has a more comprehensive set of subroutines and allows for more control by the user. Also, while some knowledge of FORTRAN is useful for purposes other than accessing IMSL, the PROTRAN syntax which the user must learn only has application to the specific software package.

Cost of the IMSL library depends on what computer it is used with. The price for the package on a VAX 780 is $3500, while for a PC it is $2050 with access to the subroutine programming code or $1500 without access. A license for the life of a VAX 780 computer is $17,000 with a $700 per year maintenance and update charge. A site license for up to 100 copies of the program for PC compatibles is $4000 each for the math, stat, and special function sub-libraries, with a $1000 maintenance/update fee per sub-library. The cost for PROTRAN is similar to that of the standard library.

METHODOLOGY

Locating the appropriate subroutine

The are 3 manuals each for the MATH/LIBRARY and the STAT/LIBRARY, and 1 for the SFUN/LIBRARY. At the end of each set of manuals are three indices of the subroutines organized as follows:

1. KWIC index - by keyword
2. GAMS index - by major subject area
3. Alphabetical index

A sample page from each of the three indices can be found in Appendix 1.

Subroutine documentation

Each routine is described concisely, and at least one example of its use is presented, with sample input and output. Algorithms are described, and references provided. Appendix 2 contains documentation from the subroutines used in the examples that follow.

Writing programs to call IMSL subroutines

The IMSL subroutines are written in FORTRAN, and are thus most easily accessed by FORTRAN programs which call them. Some computer operating systems allow the subroutines to be called by programs written in other languages. The method of doing so is peculiar to each operating system.

Very simple FORTRAN programs can be written to read data from a file,

store the information in computer memory, call IMSL subroutines to perform mathematical or statistical manipulations and write the results out to the terminal screen or a file.

In most computers, creation of a program consists of entering a mode in which text can be stored in a file, typing in the program steps, saving the file, and instructing the computer to compile and link the program, which converts the program to machine language that the computer can follow directly and allocates the appropriate computer resources. The program is then ready to run on a specified data file. A potential user must find out the instructions needed on his particular computer to store, compile, link and run a FORTRAN program.

Some FORTRAN fundamentals

Variables

Numbers are stored under variable names. Values can be assigned to a variable directly in the program, from the terminal keyboard, or read from a file.

Examples:

Assigning values to variables:

```
A=3.5
B=14.7
```

Reading values from the terminal keyboard:

```
PRINT*,"ENTER SUBJECTS HEIGHT IN METERS"
READ*,HEIGHT
```

Arrays

An array is a storage place for numbers. It can have one, two or more dimensions. Each array must be declared at the beginning of the program.

Examples:

```
REAL A,VARS(8),VALUES(2,6)
```

A is a variable which can hold one numerical value at a time. VARS is a one-dimensional array of length 8 that can store 8 numbers. The contents of VARS might be:

10.2	13.5	14.8	34.7	89.4	67.2	78.2	36.8

VALUES is a two-dimensional array with 2 rows and 6 columns, that can store 2 x 6 = 12 numbers. The contents of VALUES might be:

56.7	19.2	24.0	13.2	45.8	95.1
24.5	12.9	97.3	64.9	32.9	74.5

Individual elements within an array are specified by numbers within parentheses following the array name. For example VARS(4) is the fourth element of array VARS and has the value 34.7. VALUES(2,5) equals 32.9 since the element in row two, column five of VALUES is specified.

Format Statements

The format statement specifies how numbers are to be read from or written to a file. The most commonly used format is the F type.

Example:

```
      READ(1,20)A,B
20 FORMAT(F4.1,F5.2)
```

The first line assigns to variables A and B numbers it reads from file 1 according to the format statement on the line labelled 20, which specifies that the two numbers will be read from the current line in the file. The first number will consist of the first 4 characters from the current line in the file, with a decimal point placed to the left of the single rightmost digit. The second number will contain the next 5 characters from the current file line, with a decimal point to the left of the rightmost 2 digits. When a file is opened, the current file line is the first line in the file. After each read statement, the current file line is incremented by one.

DO Loops

In a DO Loop, a set of instructions is used repeatedly for a specified number of times.

Example:

```
      DO 30 I=1,100
      READ(1,20)KILOMETERS(I)
30 MILES(I)=KILOMETERS(I)*.6214
```

The 30 after the word 'DO' indicates the label of the last line of the loop. I is the loop counter whose value is incremented by 1 at the end of each

pass through the loop. This loop reads elements 1 through 100 of array KILOMETERS from a file and assigns corresponding values to elements 1 through 100 of array MILES.

Integers and real numbers

Integers are whole numbers used for counting. They cannot have fractional parts or decimal points. Examples of integers are 1, 3, 11, 506, and 10786. Real numbers can express gradations between whole numbers and have decimal points. Examples of real numbers are 13.5, 22., 1050.525 and 0.148.

It must be specified in FORTRAN which variable names refer to integers and which to real numbers. The specification can be in two ways. The first involves the first letter of the variable name. If the name starts with I,J,K,L,M or N, the variable is taken to be an integer unless specified otherwise. A variable name with any other starting letter is assumed to be a real number unless specified otherwise. Variables specifically declared as integers or real numbers override the starting letter convention.

Example of variable declaration:

```
INTEGER COUNT, FLAG
REAL KILOGRAMS, MINUTES
```

Thus the variables COUNT and FLAG are integers even though they start with letters other than i,j,k,l,m or n while KILOGRAMS and MINUTES are real numbers even though they start with letters other than i through n.

Calling subroutines

A subroutine is a program segment that can be called to perform a task. An example of a subroutine call is:

```
CALL AREAC(R,AREA)
```

This particular subroutine calculates the area of a circle when a radius is provided. Note that R is a variable that must be input, while AREA is calculated by the subroutine.

Example:

```
RAD=5.
CALL AREAC(RAD,A)
PRINT*, A
```

The area of the circle of radius 5.0 will be printed. Note that names of the arguments in the subroutine call don't matter, but order and type of the arguments must be as specified. Actual numbers as opposed to variable names may be used directly in the subroutine call if desired:

```
CALL AREAC(5.0,A)
```

Single and double precision

Each computer has a certain number of digits past the decimal point to which a real number may be considered accurate. Smaller computers usually have fewer digits of precision than do larger computers. Single precision is adequate for most mathematical operations. Yet sometimes, particularly where a

lot of repetitive mathematical operations may magnify errors, the standard degree of precision is not adequate. When it isn't, computers can be asked to define real numbers in double precision, where twice the normal computer memory is allocated per number stored, increasing accuracy considerably. Personal computers, in which standard numerical precision is considerably less than in mainframes, would more often require specification of double precision.

An example of defining variables as double precision on the VAX 780 computer:

```
DOUBLE PRECISION HEIGHT, WEIGHT, VARS(100)
```

Most IMSL subroutines come in both single and double precision versions. In such cases, on the description page of the subroutine, the single and double precision titles are separated by a slash, with the double precision name coming second and beginning with a "D", as in:

```
LFSQH/DLFSQH
```

The following is an example of a call to a single precision subroutine:

```
call LFSQH( N, FAC, LDFAC, NCODA, B, X )
```

For double precision, real numbers must be defined as double precision at the start of the program:

```
DOUBLE PRECISION FAC, LDFAC, B, X
```

The double precision version of the subroutine is then called when needed:

```
CALL DLFSQH( N, FAC, LDFAC, NCODA, B, X )
```

* Note - only the real numbers, and not the integers, are defined as double precision. Integers are used for counting, and have no decimal point (e.g. 1, 12, 4055). By default in FORTRAN all variable names beginning with I, J, K, L, M or N specify integers. Variable names beginning with any other letter specify real numbers, which have decimal points (e.g. 20.25, .0034, 10000.), and can theoretically express infinite gradation between whole numbers.

An example of using mathematical subroutines in biomechanics

Figure 1 is a plot of torque produced throughout a range of human joint motion on a dynamometer. Actual data points are indicated by the squares, which are joined by straight lines. Figure 2 shows the area under each straight line segment divided into a rectangle and a triangle. Using standard formulas to get the areas of all the rectangles and triangles, the total area under the curve adds up to 27,615 units.

A problem with the above approach is that joining the data points by straight lines does not adequately represent the smooth gradation of human torque capability over a range of joint motion. An IMSL subroutine can be used to draw a smooth curve through the data points. The following program reads the torque and joint angle corresponding to each point on the graph, calls IMSL cubic spline smoothing subroutines and writes a set of points corresponding to the smooth curve to a new file. Another IMSL subroutine calculates the area under the curve.The raw and smoothed data points can be plotted with any standard plotting package or by another IMSL routine using line-printer graphics.

Figure 3 shows the raw data points, and the cubic spline derived smooth curve plotted from the x and y coordinates in the output file. The automatic spline subroutine used in the program chooses the degree of smoothing based on statistical considerations. Note that the points are not necessarily intersected by the smoothing curve. The area calculated under the smooth curve is not the same as that calculated using rectangles and triangles. With other data distribution shapes, the discrepancy would be even greater. The smooth curve is more representative of most natural phenomena than is the series of line segments.

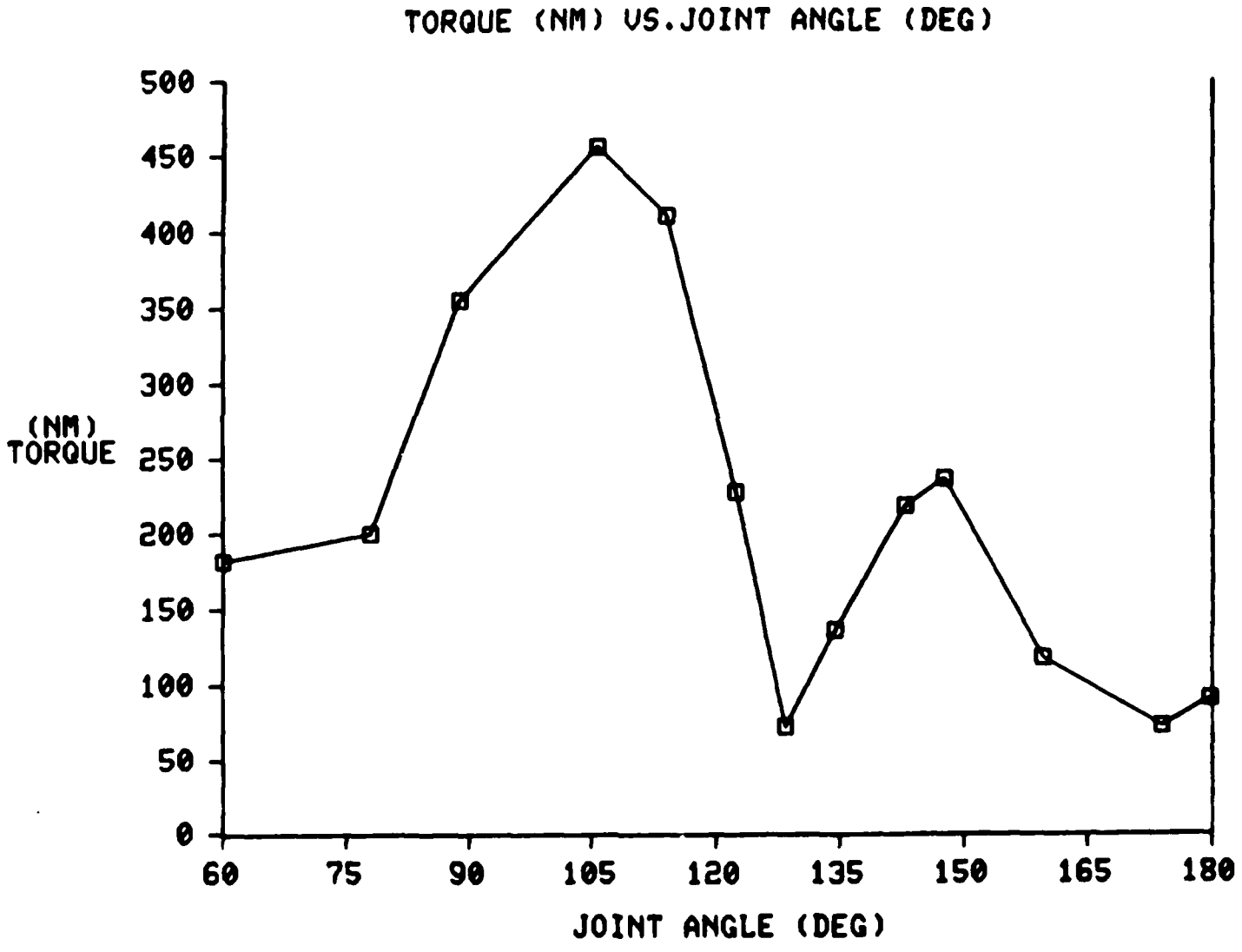

Figure 1 - Raw data points connected by straight lines

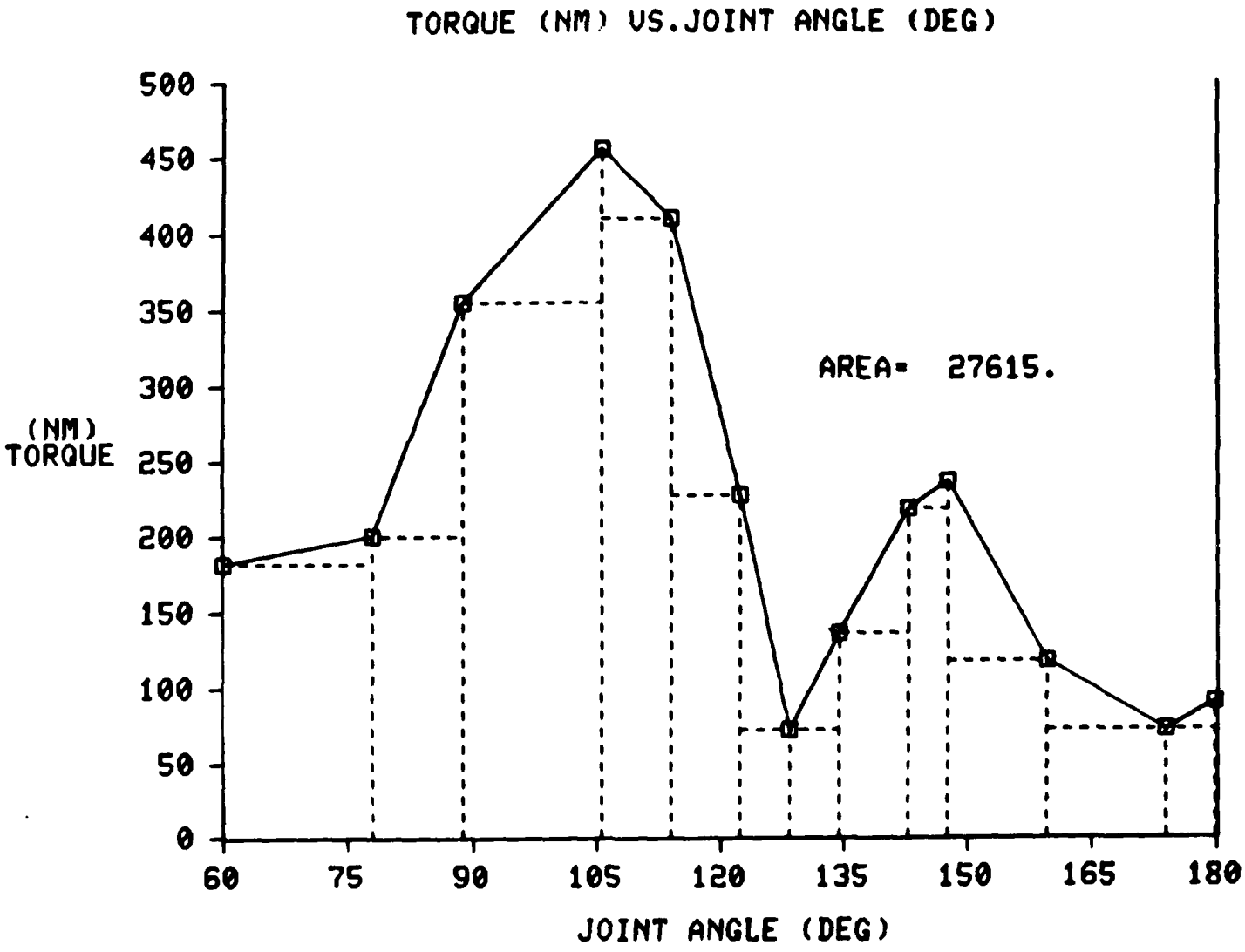

Figure 2 - Area calculation using rectangles and triangles

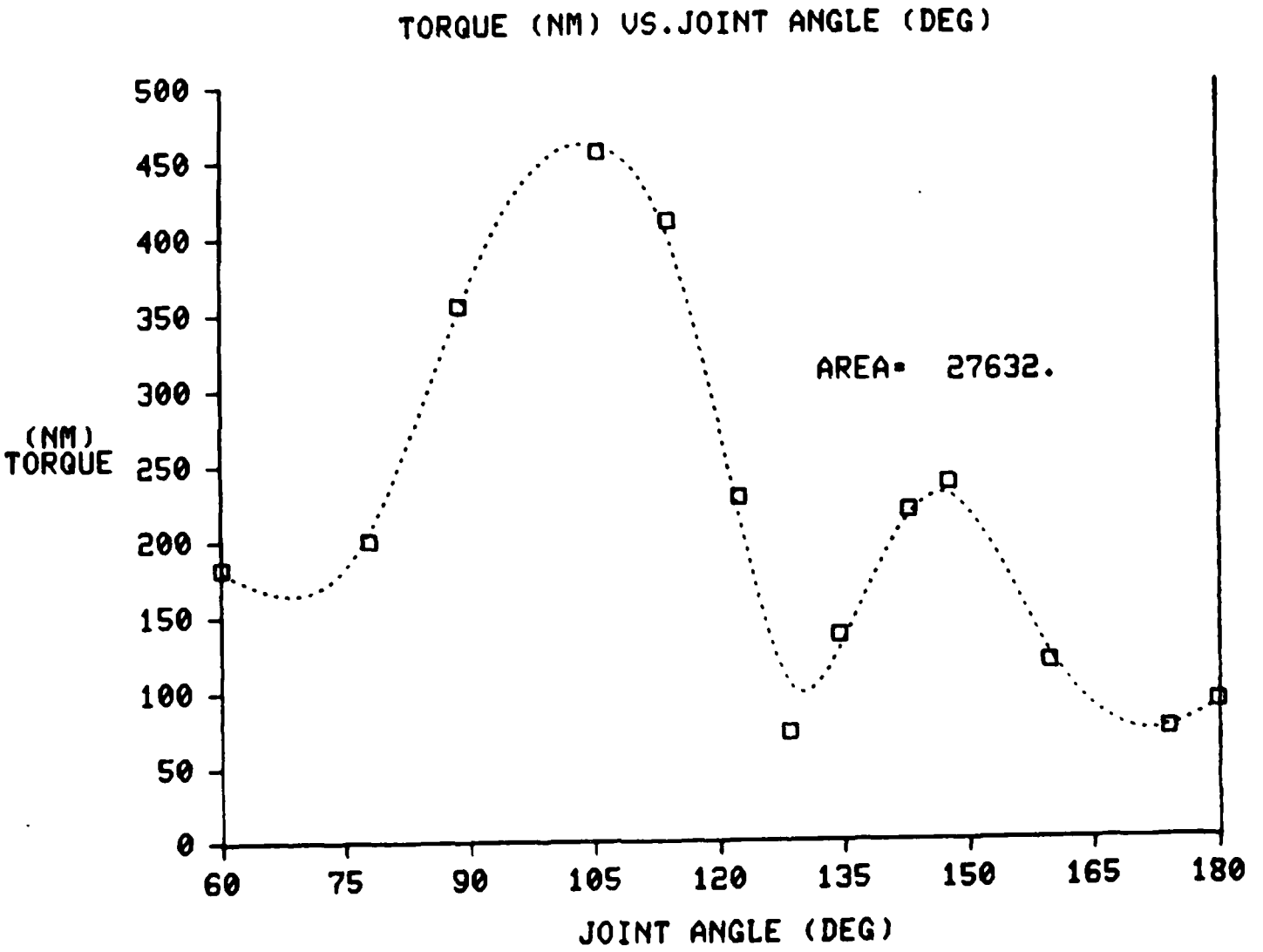

Figure 3 - Raw data points with automatic cubic spline curve superimposed

It should be noted that program line numbers have been placed on the left to facilitate discussion and are not to be typed in as part of the FORTRAN program. A line-by-line discussion of the program follows the listing.

```
 1       REAL X(13),Y(13),XNEW(121),YNEW(121),BREAK(13),CSCOEF(4,13)
 2       OPEN(1,FILE='POINTS.DAT',STATUS='OLD')
 3       OPEN(2,FILE='SMOOTHED.DAT',STATUS='NEW')
 4       DO 10 I=1,13
 5    10 READ(1,20)X(I),Y(I)
 6    20 FORMAT(2F6.2)
 7       CALL CSSCV(13,X,Y,2,BREAK,CSCOEF)
 8       AREA=CSITG(60.,180.,12,BREAK,CSCOEF)
 9       WRITE(2,30)AREA
10    30 FORMAT('AREA = ',F7.0)
11       ICOUNT=0
12       DO 40 IANGLE=60,180
13       ICOUNT=ICOUNT+1
14       XNEW(ICOUNT)=IANGLE
15       YNEW(ICOUNT)=CSVAL(XNEW(I),12,BREAK,CSCOEF)
16    40 WRITE(2,20)XNEW(I),YNEW(I)
17       CLOSE(1)
18       CLOSE(2)
19       STOP
20       END
```

line explanation

1 Declaration is made of the arrays of real numbers. X and Y are the abscissae and ordinates of the raw data points. Both are one dimensional arrays dimensioned to length 13, since there are 13 raw data points. XNEW and YNEW are arrays set up to hold points along a smooth curve calculated with the cubic spline. Both are one dimensional arrays of length 121 to hold abscissae and ordinates corresponding to the range of joint motion between 60 and 180 degrees at one degree increments. The arrays BREAK and CSCOEF are required by the IMSL subroutines used in the program. Partial documentation from the IMSL manual on cubic spline subroutines CSSCV, CSITG and CSVAL is shown in figure 4. Complete documentation on all subroutines referred to in this report can be found in appendix 2. It can be seen that when the subroutine is called, there is a list of arguments in parenthesis after the subroutine name.

2 All files from which data is read or to which data is written must be opened. This open statement assigns number 1 to the pre-existing or 'old' input file called POINTS.DAT. Any further reference to the file in the program is by its number.

3 The 'new' file created by the program, which contains the smoothed data points, is assigned number 2.

4-6 Lines 4 and 5 comprise a DO loop in which successive elements of arrays X and Y are read from file 1 according to the format statement on the line labelled 20.

7 The cubic spline subroutine is called. Values for the arguments are assigned according to the instructions in figure 4. After the

CSSCV/DCSSCV (Single/Double precision)

Purpose: Compute a smooth cubic spline approximation to noisy data using cross-validation to estimate the smoothing parameter.

Usage: CALL CSSCV (NDATA, XDATA, FDATA, IEQUAL, BREAK, CSCOEF)

Arguments

NDATA - Number of data points. (Input)
NDATA must be at least 4.

XDATA - Array of length NDATA containing the data point abscissas. (Input)
XDATA must be distinct.

FDATA - Array of length NDATA containing the data point ordinates. (Input)

IEQUAL - A flag alerting the subroutine that the data is equally spaced. (Input)
If NDATA is small (less than about 20) then IQUEAL should be set to 2.
If IEQUAL is 1 then equal spacing is assumed and the algorithm is more efficient; otherwise, unequal spacing for the XDATA vector is assumed.

BREAK - Array of length NDATA containing the breakpoints for the piecewise cubic representation. (Output)

CSCOEF - Matrix of size 4 by NDATA containing the local coefficients of the cubic pieces (Output)

CSITG/DCSITG (Single/Double precision)

Purpose: Evaluate the integral of a cubic spline.

Usage: CSITG(A, B, NINTV, BREAK, CSCOEF)

Arguments

A - Lower limit of integration. (Input)

B - Upper limit of integration. (Input)

NINTV - Number of polynomial pieces. (Input)

BREAK - Array of length NINTV+1 containing the breakpoints for the piecewise cubic representation. (Input)
BREAK must be strictly increasing

CSCOEF - Matrix of size 4 by NINTV+1 containing the local coefficients of the cubic pieces. (Input)

CSITG - Value of the integral of the spline from A to B. (Output)

CSVAL/DCSVAL (Single/Double precision)

Purpose: Evaluate a cubic spline.

Usage: CSVAL(X, NINTV, BREAK, CSCOEF)

Arguments

X - Point at which the spline is to be evaluated. (Input)

NINTV - Number of polynomial pieces. (Input)

BREAK - Array of length NINTV+1 containing the breakpoints for the piecewise cubic representation. (Input)
BREAK must be strictly increasing

CSCOEF - Matrix of size 4 by NINTV+1 containing the local coefficients of the cubic pieces. (Input)

CSVAL - Value of the polynomial at X. (Output)

Figure 4 - Arguments for subroutines CSSCV, CSITG and CSVAL

subroutine call, arrays BREAK and CSCOEF have values specifying cubic spline smoothing equations for each interval between the raw data points.

8 Area is calculated by the subroutine CSITG. Arrays BREAK and CSCOEF already contain values assigned to them by CSSCV on the previous line.

9-10 The area is written to file 2 according to the format statement on the line labelled 30.

11 The array element counter ICOUNT is initialized to 0.

12-16 This group of lines is a DO loop whose instructions are repeated as the variable IANGLE assumes successive values between 60 and 180. The elements of XNEW are assigned real number values corresponding to the integer values of IANGLE. The elements of Y are assigned values by subroutine CSVAL which calculates them using the smoothing equation coefficients determined by subroutine CSSCV. Values of XNEW and YNEW are written to file 2 according to the format indicated on the line labelled 20.

17-18 Files 1 and 2 are closed.

19-20 Standard program ending.

Sometimes the Y values corresponding to abscissae between actual data points are needed. Figure 5 compares Y values chosen by straight line and cubic spline interpolation. It can be seen that there can be considerable discrepancy between the two. Where it is known that a smooth curve is more representative of a phenomenon, spline interpolation is preferable.

In some cases automatic smoothing may not be appropriate. An example would be where the user would like the curve to pass through all the data points. In

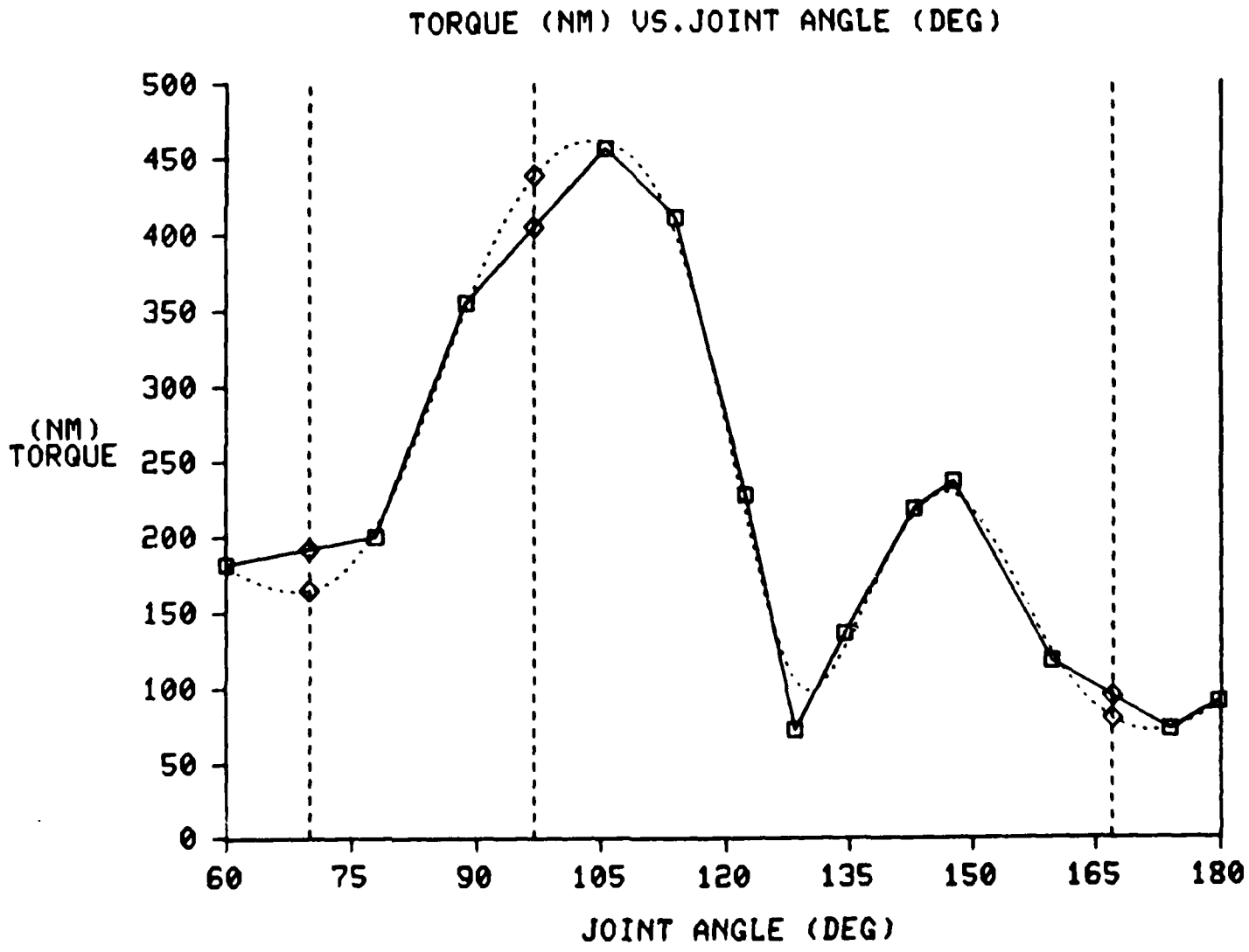

Figure 5 - Comparison of straight line and cubic spline interpolation

cases where the user would like to decide on the degree of smoothing, subroutine CSSMH should be used instead of subroutime CSSCV. The arguments for CSSMH are shown in figure 6. It can be seen that two new arguments are needed, WEIGHT and SMPAR. In Figure 7, the smallest dashes show the automatically smoothed curve using CSSCV, the mid-size dashes show a less smooth curve which stays close to all raw data points, and the large dashes show an excessively smooth curve, for which a high value for SMPAR was specified. Modifications to the program for user specified degree of smoothing include declaration, dimensioning and assigning values to array WEIGHT, reading a value for SMPAR, and calling subroutine CSSMH instead of CSSCV. All 13 elements of WEIGHT were arbitrarily assigned values of 25.

```
REAL WEIGHT(13)
DATA WEIGHT/13*25./
          .
          .
PRINT*,'ENTER THE SMOOTHING FACTOR:'
READ*,SMPAR
CALL CSSMH(13,X,Y,WEIGHT,SMPAR,BREAK,CSCOEF)
          .
          .
```

The rest of the program is as before. The 'PRINT*,' and 'READ*,' instructions allow writing a question to the terminal screen and reading the answer from the keyboard. A graphics package or routine can be used to visually check if values chosen for WEIGHT and SMPAR provide the desired degree of smoothing.

CSSMH/DCSSMH (Single/Double precision)

Purpose: Compute a smooth cubic spline approximation to noisy data.

Usage: CALL CSSMH (NDATA, XDATA, FDATA, WEIGHT, SMPAR, BREAK, CSCOEF)

Arguments

NDATA - Number of data points. (Input)
NDATA must be at least 2.

XDATA - Array of length NDATA containing the data point abscissas. (Input)
XDATA must be distinct.

FDATA - Array of length NDATA containing the data point ordinates. (Input)

WEIGHT - Array of length NDATA containing estimates of the standard deviations of FDATA. (Input)
All elements of WEIGHT must be positive.

SMPAR - A nonnegative number which controls the smoothing. (Input)
The spline function S returned is such that the sum from I=1 to NDATA of
((S(XDATA(I))-FDATA(I))/WEIGHT(I))**2
is less than or equal to SMPAR. It is recommended that SMPAR lie in the confidence interval of this sum, i.e., NDATA-SQRT(2*NDATA) .LE. SMPAR .LE. NDATA+SQRT(2*NDATA).

BREAK - Array of length NDATA containing the breakpoints for the piecewise cubic representation. (Output)

CSCOEF - Matrix of size 4 by NDATA containing the local coefficients of the cubic pieces. (Output)

Figure 6 - Arguments for subroutine CSSMH

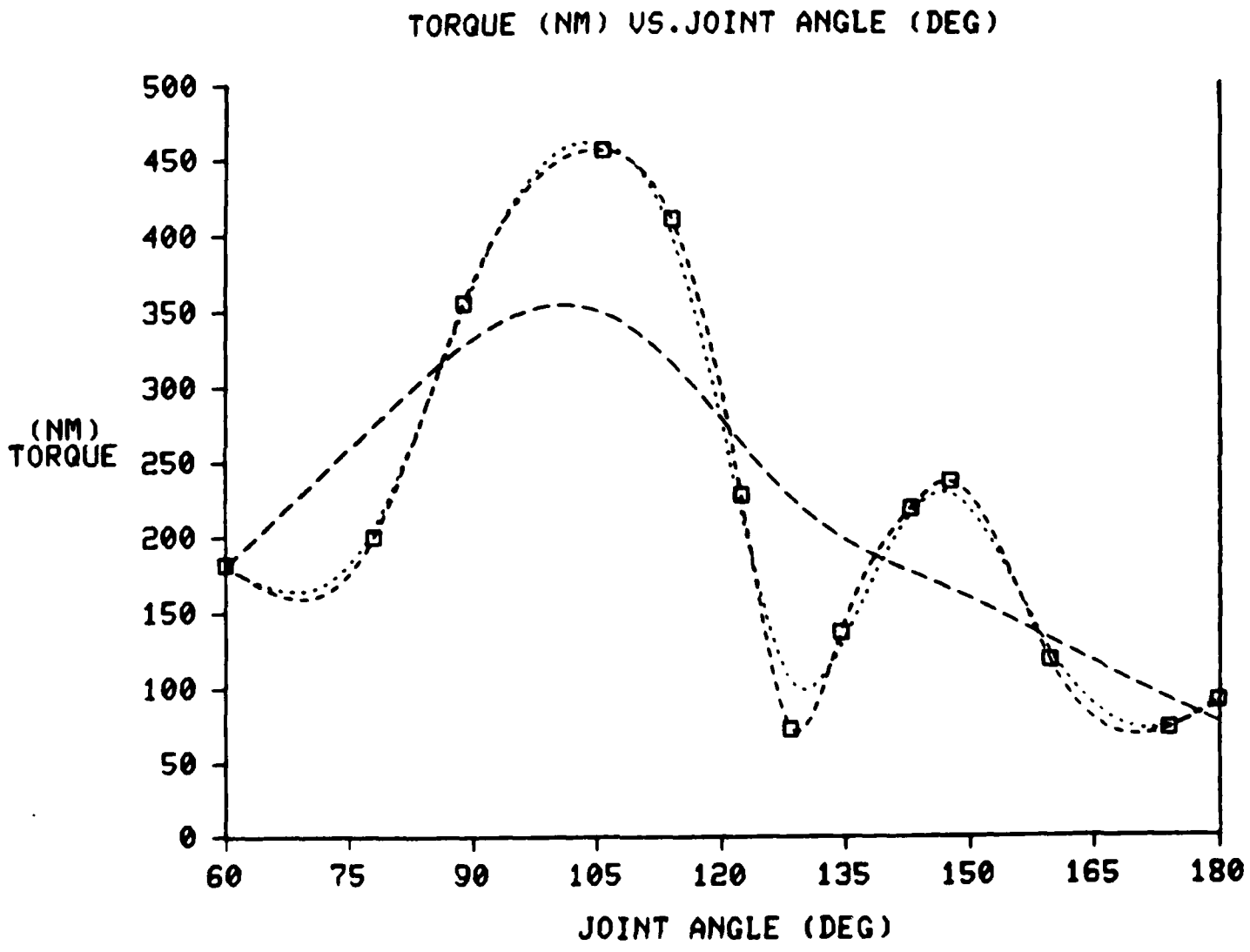

Figure 7 - Raw data points with three different spline curves superimposed

An example of using statistical subroutines in physiology

Standard linear regression derives a straight line equation to fit data such that the sum of squared deviations between the observed and predicted data points is minimized. One problem with the procedure is that an outlying point has a disproportionate amount of weight. A form of regression which is less affected by aberrant data points is one in which the sum of absolute values rather than the squared values of differences between observed and predicted points is minimized. The BMDP and SPSS statistical packages do not contain routines which perform such regression, while IMSL does. Figure 8 shows the arguments for subroutine RLINE and RLAV which respectively perform least squares and least absolute values regression.

The points plotted in figure 9 were generated by a program using a subroutine to produce uniform pseudo-random deviations about the solid straight line of slope 1.0 and intercept 0.0. The dashed lines of best fit were derived by calls to the subroutines using the least squares and least absolute value regression methods. In figure 10, a high outlier was substituted for the leftmost point of the random distribution about the Y=X line serving to increase the intercept and decrease the slope of a fitted line. It can be seen that the outlier had much greater effect on the slope and intercept obtained by the least squares method than on the slope and intercept determined by the least absolute values method, indicating that the latter method may be preferable for fitting a straight line to some data distributions. The following are essential parts of the program used to call the regression subroutines. Line-by-line notation follows.

RLINE/DRLINE (Single/Double precision)

Purpose Fit a line to a set of data points using least squares

Usage CALL RLINE (NOBS, XDATA, YDATA, BO, B1, STAT)

Arguments

NOBS - Number of observations (Input)
XDATA - Vector of length NOBS containing the x values. (Input)
YDATA - Vector of length NOBS containing the y values (Input)
BO - Estimated intercept of the fitted line (Output)
B1 - Estimated slope of the fitted line. (Output)
STAT - Vector of length 12 containing the statistics described below. (Output)

I	STAT(I)
1	Mean of XDATA
2	Mean of YDATA
3	Sample variance of XDATA
4	Sample variance of YDATA
5	Correlation
6	Estimated standard error of BO
7	Estimated standard error of B1
8	Degrees of freedom for regression
9	Sum of squares for regression
10	*Degrees of freedom for error*
11	Sum of squares for error
12	Number of (x,y) points containing NaN (not a number) as either the x or y value

---

RLAV/DRLAV (Single/Double precision)

Purpose Fit a multiple linear regression model using the least absolute values criterion.

Usage CALL RLAV (NOBS, NCOL, X, LDX, INTCEP, IIND, INDIND, IRSP, B, IRANK, SAE, ITER, NRMISS)

Arguments

NOBS - Number of observations (Input)
NCOL - Number of columns in X (Input)
X - NOBS by NCOL matrix containing the data (Input)
LDX - Leading dimension of X exactly as specified in the dimension statement in the calling program (Input)
INTCEP - Intercept option. (Input)

INTCEP	Action
0	An intercept is not in the model
1	An intercept is in the model

IIND - Independent variable option (Input)
The absolute value of IIND is the number of independent (explanatory) variables. The sign of IIND specifies the following options

IIND	Meaning
LT 0	The data for the -IIND independent variables are given in the first -IIND columns of X
GT 0	The data for the IIND independent variables are in the columns of X whose column numbers are given by the elements of INDIND
EQ 0	There are no independent variables

The regressors are the constant *regressor* (if *INTCEP* = 1) and the independent variables
INDIND - Index vector of length IIND containing the column numbers of X that are the independent (explanatory) variables (Input, if IIND is positive)
If IIND is negative, INDIND is not referenced and can be a vector of length one
IRSP - Column number IRSP of X contains the data for the response (dependent) variable (Input)
B - Vector of length INTCEP + IABS(IIND) containing a LAV solution for the regression coefficients (Output)
If INTCEP = 1, B(1) contains the intercept estimate B(INTCEP+I) contains the coefficient estimate for the I-th independent variable
IRANK - Rank of the matrix of regressors (Output)
If IRANK is less than INTCEP + IABS(IIND), linear dependence of the regressors was declared
SAE - Sum of the absolute values of the errors (Output)
ITER - Number of iterations performed (Output)
NRMISS - Number of rows of data containing NaN (not a number) for the dependent or independent variables (Output)
If a row of data contains NaN for any of these variables that row is excluded from the computations

Figure 8 - Arguments for subroutines RLINE and RLAV

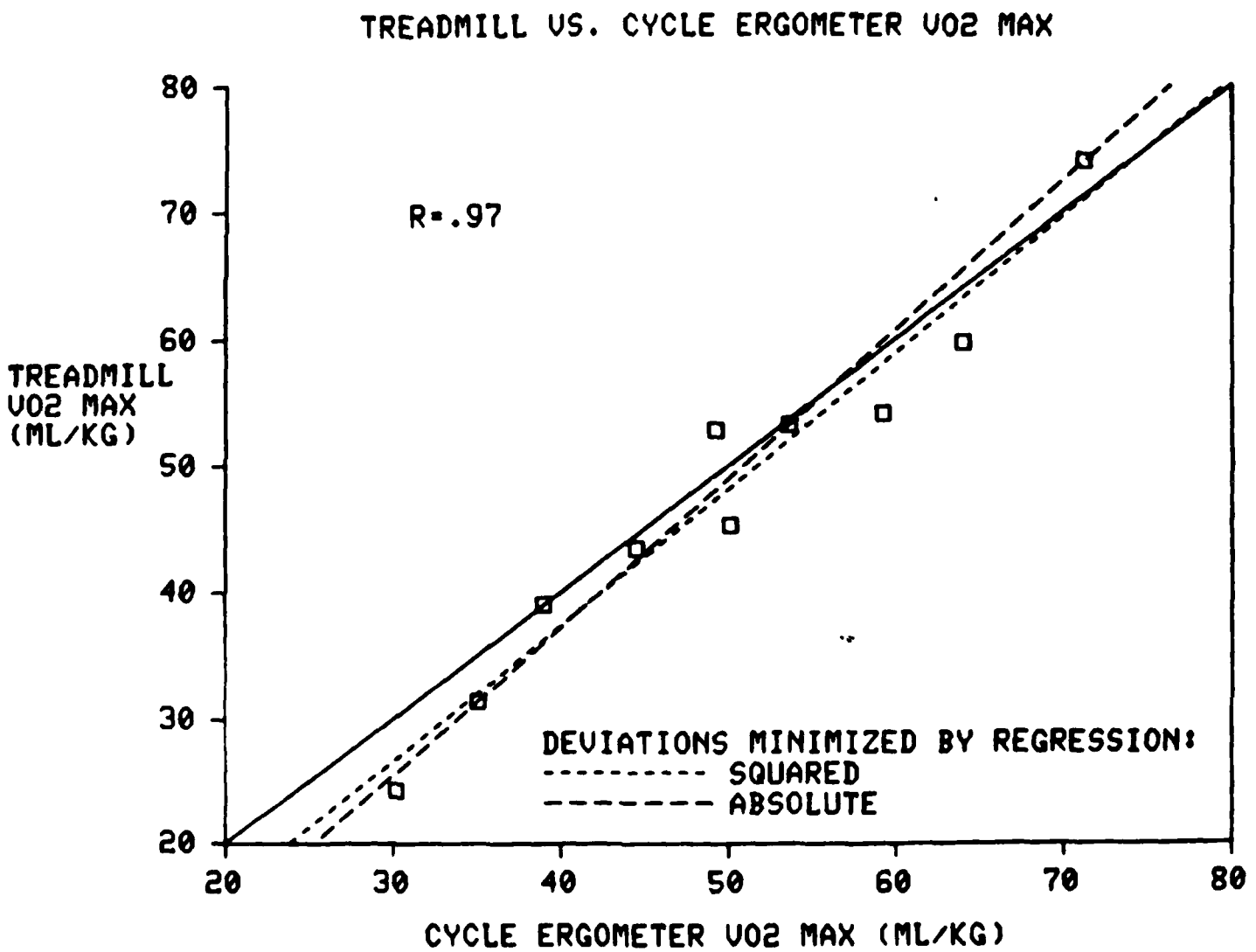

Figure 9 - Least squares and absolute value regression on a set of data points

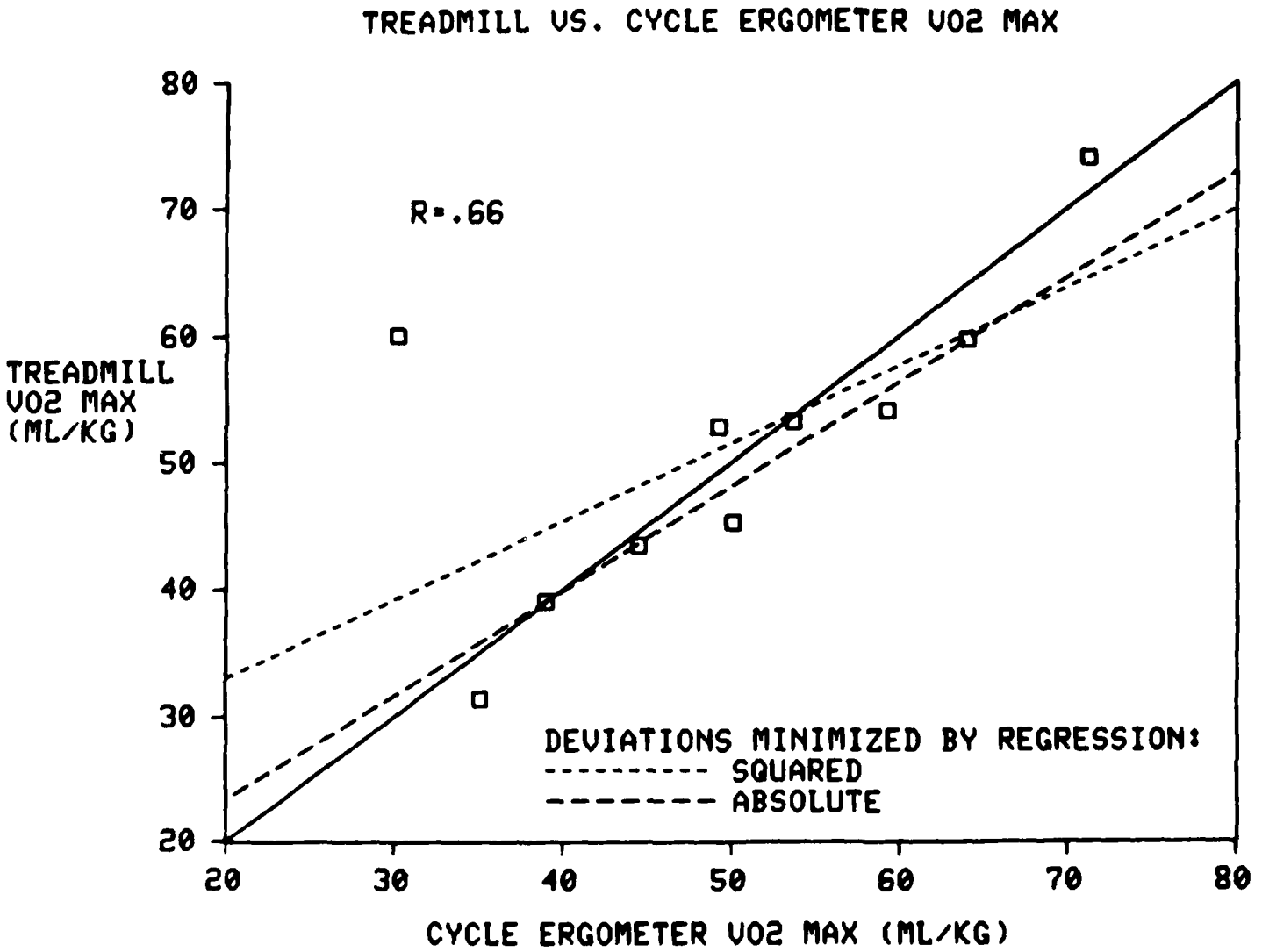

Figure 10 - The effects of an outlying point on both types of regression

```
1       REAL X(10),Y(10),STAT(12),ARREI(10,2),B(2)
                        .
                        .
2       CALL RLINE(10,X,Y,B0,B1,STAT)
3       DO 15 I=1,10
4       ARREI(I,1)=X(I)
5    15 ARREI(I,2)=Y(I)
6       CALL RLAV(10,2,ARREI,10,1,-1,2,B,IRANK,SAE,ITER,NRMISS)
                        .
                        .
```

line explanation

1 Arrays X and Y contain the data point abscissae and ordinates. Array STAT holds statistical output from subroutine RLINE. Array ARREI is a two dimensional array or matrix created by the program to satisfy the input requirements of subroutine RLAV. Array B stores results from RLAV.

2 Subroutine RLINE is called to perform least squares linear regression. After the line is executed B0 contains the intercept and B1 the slope of the fitted line. STAT(5) contains the correlation coefficient. Information can be output to a file or the terminal screen.

3-5 DO Loop 15 is used to write X and Y data to an array named ARREI to satisfy the input requirements of subroutine RLAV.

6 The subroutine is called as specified in the instructions. After the line is executed, B(1) contains the intercept and B(2) contains the slope of the fitted line.

REFERENCES

1. MATH/LIBRARY - FORTRAN Subroutines for Mathematical Applications. IMSL. Houston, Texas. 1987.
2. STAT/LIBRARY - FORTRAN Subroutines for Statistical Analysis. IMSL. Houston, Texas. 1987.

DISCLAIMER STATEMENT

The views, opinions, and/or findings contained in this report are those of the author and should not be construed as official Department of the Army position, policy, or decision.

APPENDIX 1

IMSL Indices

Keyword in Context Index

GAMS Index

The following index lists routines in MATH/LIBRARY based on the tree-structured GAMS classification scheme (Boisvert, Howe, and Kahaner 1984). Only the leaves and branches of the GAMS classification scheme that contain MATH/LIBRARY routines are included here. This index uses GAMS version 1.3, which contains some additions to and modifications of version 1.2 (Boisvert, Howe, and Kahaner 1985). An asterisk (*) following a subclass description indicates that this subclass and/or its substructure have been added or modified in version 1.3. The page number for the documentation and the purpose of the routine appear alongside the routine name.

The first level of the full GAMS classification scheme contains the following major subject areas:

A. Arithmetic, Error Analysis
B. Number Theory
C. Elementary and Special Functions
D. Linear Algebra
E. Interpolation
F. Solution of Nonlinear Equations
G. Optimization
H. Differentiation and Integration
I. Differential and Integral Equations
J. Integral Transforms
K. Approximation
L. Statistics, Probability
M. Simulation, Stochastic Modeling
N. Data Handling
O. Symbolic Computation
P. Computational Geometry
Q. Graphics
R. Service Routines
S. Software Development Tools

There are seven levels in the classification scheme. Classes in the first level are identified by a capital letter as is given above. Subclasses are identified by alternating letter-and-number combinations. A single letter (a-z) is used within the odd-numbered levels. A number (1-26) is used within the even-numbered levels.

References

Boisvert, Ronald F., Sally E. Howe, and David K. Kahaner (1984), *Guide to Available Mathematical Software*, U.S. Department of Commerce, National Bureau of Standards, Center for Applied Mathematics, Washington, D.C.

Boisvert, Ronald F., Sally E. Howe, and David K. Kahaner (1985), GAMS: A framework for the management of scientific software, *ACM Transactions on Mathematical Software*, **11**, 313-355.

Alphabetical Index of Routines

Note: Routines marked with an asterisk (*) are not user callable, but the names are reserved. See 'Reserved Names' in the Reference Material for additional details.

APPENDIX 2

Documentation of IMSL subroutines used in examples

CSSCV/DCSSCV (Single/Double precision)

Purpose: Compute a smooth cubic spline approximation to noisy data using cross-validation to estimate the smoothing parameter.

Usage: CALL CSSCV (NDATA, XDATA, FDATA, IEQUAL, BREAK, CSCOEF)

Arguments

NDATA - Number of data points. (Input)
NDATA must be at least 4.

XDATA - Array of length NDATA containing the data point abscissas. (Input)
XDATA must be distinct.

FDATA - Array of length NDATA containing the data point ordinates. (Input)

IEQUAL - A flag alerting the subroutine that the data is equally spaced. (Input)
If NDATA is small (less than about 20) then IQUEAL should be set to 2.
If IEQUAL is 1 then equal spacing is assumed and the algorithm is more efficient; otherwise, unequal spacing for the XDATA vector is assumed.

BREAK - Array of length NDATA containing the breakpoints for the piecewise cubic representation. (Output)

CSCOEF - Matrix of size 4 by NDATA containing the local coefficients of the cubic pieces. (Output)

Remark

Automatic workspace usage is

```
CSSCV    8*NDATA units if IEQUAL is 1, or
         7*NDATA + 3*NDATA**2 units otherwise, or
DCSSCV   15*NDATA units if IEQUAL is 1, or
         13*NDATA + 6*NDATA**2 units otherwise.
```

Workspace may be explicitly provided, if desired, by use of C2SCV/DC2SCV. The reference is

```
CALL C2SCV (NDATA, XDATA, FDATA, IEQUAL, BREAK, CSCOEF,
            WK, SDWK, IPVT)
```

The additional arguments are as follows:

WK - Work array of length 6*NDATA if IEQUAL is 1 or 5*NDATA+3*NDATA**2 otherwise.

SDWK - Work array of length NDATA to hold the smoothed data.

IPVT - Work array of length NDATA.

Algorithm

CSSCV is designed to produce a C^2 cubic spline approximation to a data set in which the function values are noisy. This spline is called a *smoothing spline*. It is a natural cubic spline with knots at all the data abscissas x = XDATA but it does *not* interpolate the data (x_i, f_i). The smoothing spline S_σ is the unique C^2 function which minimizes

$$\int_a^b S''_\sigma(x)^2\,dx$$

subject to the constraint

$$\sum_{i=1}^{N} |S_\sigma(x_i) - f_i|^2 \leq \sigma,$$

where σ is the smoothing parameter and N = NDATA. The reader should consult Reinsch (1967) for more information concerning smoothing splines.

The IMSL subroutine CSSMH solves the above problem when the user provides the smoothing parameter σ. This routine attempts to find the 'optimal' smoothing parameter using the statistical technique known as cross-validation. This means that (in a very rough sense) one chooses the value of σ so that the smoothing spline (S_σ) best approximates the value of the data at x_i, if it is computed using all the data *except* the i-th; this is true for all $i = 1, \ldots, N$. For more information on this topic we refer the reader to Craven and Wahba (1979).

This routine has a switch, IEQUAL, that allows the user to take advantage of the fact that the entries in XDATA are equally spaced. This switch, when it is appropriate to use, can result in more efficient execution.

Example

In this example function values are computed and are contaminated by adding a small 'random' amount. CSSCV is used to try to reproduce the original, uncontaminated data.

```
      INTEGER    NDATA
      PARAMETER  (NDATA=300)
C
      INTEGER    I, IEQUAL, NOUT
      REAL       BREAK(NDATA), CSCOEF(4,NDATA), CSVAL, ERROR, F,
     &           FDATA(NDATA), FLOAT, FVAL, RNUNF, SVAL, X,
     &           XDATA(NDATA), XT
      INTRINSIC  FLOAT
      EXTERNAL   CSSCV, CSVAL, RNSET, RNUNF, UMACH
C
      F(X) = 1.0/(.1+(3.0*(X-1.0))**4)
C
      CALL UMACH (2, NOUT)
C                                    Set up a grid
      DO 10  I=1, NDATA
         XDATA(I) = 3.0*(FLOAT(I-1)/FLOAT(NDATA-1))
```

```
         FDATA(I) = F(XDATA(I))
   10 CONTINUE
C                                    Introduce noise on [-.5,.5]
C                                    Contaminate the data
      CALL RNSET (1234579)
      DO 20  I=1, NDATA
         FDATA(I) = FDATA(I) + 2.0*RNUNF() - 1.0
   20 CONTINUE
C
C                                    Set IEQUAL=1 for equally spaced data
      IEQUAL = 1
C                                    Smooth data
      CALL CSSCV (NDATA, XDATA, FDATA, IEQUAL, BREAK, CSCOEF)
C                                    Print results
      WRITE (NOUT,99999)
      DO 30  I=1, 10
         XT    = 90.0*(FLOAT(I-1)/FLOAT(NDATA-1))
         SVAL  = CSVAL(XT,NDATA-1,BREAK,CSCOEF)
         FVAL  = F(XT)
         ERROR = SVAL - FVAL
         WRITE (NOUT,'(4F15.4)') XT, FVAL, SVAL, ERROR
   30 CONTINUE
99999 FORMAT (12X, 'X', 9X, 'Function', 7X, 'Smoothed', 10X,
     &       'Error')
      END
```

Output

X	Function	Smoothed	Error
.0000	.0123	.2552	.2429
.3010	.0514	.1062	.0547
.6020	.4690	.3121	-.1569
.9030	9.3311	8.9495	-.3817
1.2040	4.1611	4.6834	.5223
1.5050	.1863	.3833	.1970
1.8060	.0292	.1161	.0869
2.1070	.0082	.0654	.0571
2.4080	.0031	.0403	.0372
2.7090	.0014	-.2158	-.2172

References

Craven, Peter, and Grace Wahba (1979), Smoothing noisy data with spline functions, *Numerische Mathematik*, **31**, 377–403.

Reinsch, Christian H. (1967), Smoothing by spline functions, *Numerische Mathematik*, **10**, 177–183.

CSITG/DCSITG (Single/Double precision)

Purpose: Evaluate the integral of a cubic spline.

Usage: CSITG(A, B, NINTV, BREAK, CSCOEF)

Arguments

A - Lower limit of integration. (Input)
B - Upper limit of integration. (Input)
NINTV - Number of polynomial pieces. (Input)
BREAK - Array of length NINTV+1 containing the breakpoints for the piecewise cubic representation. (Input) BREAK must be strictly increasing.
CSCOEF - Matrix of size 4 by NINTV+1 containing the local coefficients of the cubic pieces. (Input)
CSITG - Value of the integral of the spline from A to B. (Output)

Algorithm

CSITG evaluates the integral of a cubic spline over an interval. It is a special case of the routine PPITG, which evaluates the integral of a piecewise polynomial. (A cubic spline is a piecewise polynomial of order 4.)

Example

This example computes a cubic spline interpolant to the function x^2 using CSINT and evaluates its integral over the intervals [0.. .5] and [0.. 2.]. Since CSINT uses the not-a-knot condition, the interpolant reproduces x^2; hence the integral values are 1/24 and 8/3, respectively.

```
      INTEGER    NDATA
      PARAMETER  (NDATA=10)
C
      INTEGER    I, NINTV, NOUT
      REAL       A, B, BREAK(NDATA), CSCOEF(4,NDATA), CSITG, ERROR,
     &           EXACT, F, FDATA(NDATA), FI, FLOAT, VALUE, X,
     &           XDATA(NDATA)
      INTRINSIC  FLOAT
      EXTERNAL   CSINT, CSITG, UMACH
C                                Define function and integral
      F(X)  = X*X
      FI(X) = X*X*X/3.0
C                                Set up a grid
      DO 10  I=1, NDATA
         XDATA(I) = FLOAT(I-1)/FLOAT(NDATA-1)
         FDATA(I) = F(XDATA(I))
```

```
   10 CONTINUE
C                                  Compute cubic spline interpolant
      CALL CSINT (NDATA, XDATA, FDATA, BREAK, CSCOEF)
C                                  Compute the integral of F over
C                                  [0.0,0.5]
      A     = 0.0
      B     = 0.5
      NINTV = NDATA - 1
      VALUE = CSITG(A,B,NINTV,BREAK,CSCOEF)
      EXACT = FI(B) - FI(A)
      ERROR = EXACT - VALUE
C                                  Get output unit number
      CALL UMACH (2, NOUT)
C                                  Print the result
      WRITE (NOUT,99999) A, B, VALUE, EXACT, ERROR
C                                  Compute the integral of F over
C                                  [0.0,2.0]
      A     = 0.0
      B     = 2.0
      VALUE = CSITG(A,B,NINTV,BREAK,CSCOEF)
      EXACT = FI(B) - FI(A)
      ERROR = EXACT - VALUE
C                                  Print the result
      WRITE (NOUT,99999) A, B, VALUE, EXACT, ERROR
99999 FORMAT (' On the closed interval (', F3.1, ',', F3.1,
     &        ') we have :', /, 1X, 'Computed Integral = ', F10.5, /,
     &        1X, 'Exact Integral    = ', F10.5, /, 1X, 'Error
     &        , '    = ', F10.6, /, /)
C
      END
```

Output

```
On the closed interval ( .0, .5) we have :
Computed Integral =     .04167
Exact Integral    =     .04167
Error             =    .000000


On the closed interval ( .0,2.0) we have :
Computed Integral =    2.66669
Exact Integral    =    2.66667
Error             =   -.000023
```

CSVAL/DCSVAL (Single/Double precision)

Purpose: Evaluate a cubic spline.

Usage: CSVAL(X, NINTV, BREAK, CSCOEF)

Arguments

X - Point at which the spline is to be evaluated. (Input)
NINTV - Number of polynomial pieces. (Input)
BREAK - Array of length NINTV+1 containing the breakpoints for the piecewise cubic representation. (Input) BREAK must be strictly increasing.
CSCOEF - Matrix of size 4 by NINTV+1 containing the local coefficients of the cubic pieces. (Input)
CSVAL - Value of the polynomial at X. (Output)

Algorithm

CSVAL evaluates a cubic spline at a given point. It is a special case of the routine PPDER, which evaluates the derivative of a piecewise polynomial. (The value of a piecewise polynomial is its zero-th derivative and a cubic spline is a piecewise polynomial of order 4.) PPDER is based on the routine PPVALU in de Boor (1978, page 89).

Example

For an example of the use of CSVAL, see IMSL routine CSINT.

Reference

de Boor, Carl (1978), *A Practical Guide to Splines*, Springer-Verlag, New York.

CSSMH/DCSSMH (Single/Double precision)

Purpose: Compute a smooth cubic spline approximation to noisy data.

Usage:
```
CALL CSSMH (NDATA, XDATA, FDATA, WEIGHT, SMPAR, BREAK,
            CSCOEF)
```

Arguments

NDATA - Number of data points. (Input)
NDATA must be at least 2.

XDATA - Array of length NDATA containing the data point abscissas. (Input)
XDATA must be distinct.

FDATA - Array of length NDATA containing the data point ordinates. (Input)

WEIGHT - Array of length NDATA containing estimates of the standard deviations of FDATA. (Input)
All elements of WEIGHT must be positive.

SMPAR - A nonnegative number which controls the smoothing. (Input)
The spline function S returned is such that the sum from I=1 to NDATA of ((S(XDATA(I))-FDATA(I))/WEIGHT(I))**2 is less than or equal to SMPAR. It is recommended that SMPAR lie in the confidence interval of this sum, i.e., NDATA-SQRT(2*NDATA) .LE. SMPAR .LE. NDATA+SQRT(2*NDATA).

BREAK - Array of length NDATA containing the breakpoints for the piecewise cubic representation. (Output)

CSCOEF - Matrix of size 4 by NDATA containing the local coefficients of the cubic pieces. (Output)

Remarks

1. Automatic workspace usage is

   CSSMH 9*NDATA+5 units, or
   DCSSMH 17*NDATA+10 units.

   Workspace may be explicitly provided, if desired, by use of C2SMH/DC2SMH. The reference is

   ```
   CALL C2SMH (NDATA, XDATA, FDATA, WEIGHT, SMPAR, BREAK,
               CSCOEF, WK, IWK)
   ```

   The additional arguments are as follows:

   WK - Work array of length 8*NDATA+5.

   IWK - Work array of length NDATA.

2. Informational error
   Type Code
   3 1 The maximum number of iterations has been reached. The best approximation is returned.

3. The cubic spline can be evaluated using CSVAL; its derivative can be evaluated using CSDER.

Algorithm

CSSMH is designed to produce a C^2 cubic spline approximation to a data set in which the function values are noisy. This spline is called a *smoothing spline.* It is a natural cubic spline with knots at all the data abscissas x = XDATA, but it does *not* interpolate the data (x_i, f_i). The smoothing spline S is the unique C^2 function which minimizes

$$\int_a^b S''(x)^2\,dx$$

subject to the constraint

$$\sum_{i=1}^{N} \left| \frac{S(x_i) - f_i}{w_i} \right|^2 \leq \sigma,$$

where w = WEIGHT, σ = SMPAR is the smoothing parameter, and N = NDATA.

Recommended values for σ depend on the weights w. If an estimate for the standard deviation of the error in the value f_i is available, then w_i should be set to this value and the smoothing parameter σ should be chosen in the confidence interval corresponding to the left side of the above inequality. That is,

$$N - \sqrt{2N} \leq \sigma \leq N + \sqrt{2N}.$$

CSSMH is based on an algorithm of Reinsch (1967). This algorithm is also discussed in de Boor (1978, pages 235-243).

Example

In this example, function values are contaminated by adding a small 'random' amount to the correct values. CSSMH is used to approximate the original, uncontaminated data.

```
      INTEGER    NDATA
      PARAMETER  (NDATA=300)
C
      INTEGER    I, NOUT
      REAL       BREAK(NDATA), CSCOEF(4,NDATA), CSVAL, ERROR, F,
     &           FDATA(NDATA), FLOAT, FVAL, RNUNF, SDEV, SMPAR, SQRT,
     &           SVAL, WEIGHT(NDATA), X, XDATA(NDATA), XT
      INTRINSIC  FLOAT, SQRT
      EXTERNAL   CSSMH, CSVAL, RNSET, RNUNF, SSET, UMACH
```

```
C
      F(X) = 1.0/(.1+(3.0*(X-1.0))**4)
C                                     Set up a grid
      DO 10  I=1, NDATA
         XDATA(I) = 3.0*(FLOAT(I-1)/FLOAT(NDATA-1))
         FDATA(I) = F(XDATA(I))
   10 CONTINUE
C                                     Set the random number seed
      CALL RNSET (1234579)
C                                     Contaminate the data
      DO 20  I=1, NDATA
         FDATA(I) = FDATA(I) + 2.0*RNUNF() - 1.0
   20 CONTINUE
C                                     Set the WEIGHT vector
      SDEV = 1.0/SQRT(3.0)
      CALL SSET (NDATA, SDEV, WEIGHT, 1)
      SMPAR = NDATA
C                                     Smooth the data
      CALL CSSMH (NDATA, XDATA, FDATA, WEIGHT, SMPAR, BREAK, CSCOEF)
C                                     Get output unit number
      CALL UMACH (2, NOUT)
C                                     Write heading
      WRITE (NOUT,99999)
C                                     Print 10 values of the function.
      DO 30  I=1, 10
         XT    = 90.0*(FLOAT(I-1)/FLOAT(NDATA-1))
C                                     Evaluate the spline
         SVAL  = CSVAL(XT,NDATA-1,BREAK,CSCOEF)
         FVAL  = F(XT)
         ERROR = SVAL - FVAL
         WRITE (NOUT,'(4F15.4)') XT, FVAL, SVAL, ERROR
   30 CONTINUE
C
99999 FORMAT (12X, 'X', 9X, 'Function', 7X, 'Smoothed', 10X,
     &        'Error')
      END
```

Output

X	Function	Smoothed	Error
.0000	.0123	.1119	.0995
.3010	.0514	.0646	.0131
.6020	.4690	.2972	-.1718
.9030	9.3311	8.7022	-.6290
1.2040	4.1611	4.7888	.6277
1.5050	.1863	.2717	.0855
1.8060	.0292	.1408	.1116
2.1070	.0082	.0826	.0743
2.4080	.0031	.0076	.0044

RLINE/DRLINE (Single/Double precision)

Purpose: Fit a line to a set of data points using least squares.

Usage: CALL RLINE (NOBS, XDATA, YDATA, B0, B1, STAT)

Arguments

NOBS - Number of observations. (Input)
XDATA - Vector of length NOBS containing the x values. (Input)
YDATA - Vector of length NOBS containing the y values. (Input)
B0 - Estimated intercept of the fitted line. (Output)
B1 - Estimated slope of the fitted line. (Output)
STAT - Vector of length 12 containing the statistics described below. (Output)

I	STAT(I)
1	Mean of XDATA
2	Mean of YDATA
3	Sample variance of XDATA
4	Sample variance of YDATA
5	Correlation
6	Estimated standard error of B0
7	Estimated standard error of B1
8	Degrees of freedom for regression
9	Sum of squares for regression
10	Degrees of freedom for error
11	Sum of squares for error
12	Number of (x,y) points containing NaN (not a number) as either the x or y value

Remark

Informational error

Type	Code	
4	1	Each (x,y) point contains NaN (not a number). There are no valid data.

Keyword: Simple linear regression

Algorithm

Subroutine RLINE fits a line to a set of (x, y) data points using the method of least squares. Draper and Smith (1981, pages 1–69) discuss the method. The fitted model is

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x,$$

where $\hat{\beta}_0$ (stored in B0) is the estimated intercept and $\hat{\beta}_1$ (stored in B1) is the estimated slope. In addition to the fit, RLINE produces some summary statistics, including the means, sample variances, correlation, and the error (residual) sum of squares. The estimated standard errors of $\hat{\beta}_0$ and $\hat{\beta}_1$ are computed under the simple linear regression model. The errors in the model are assumed to be uncorrelated and with constant variance.

If the x values are all equal, the model is degenerate. In this case, RLINE sets $\hat{\beta}_1$ to zero and $\hat{\beta}_0$ to the mean of the y values.

Example

This example fits a line to a set of data discussed by Draper and Smith (1981, Table 1.1, pages 9–33). The response y is the amount of steam used per month (in pounds) and the independent variable x is the average atmospheric temperature (in degrees Fahrenheit).

```
C                                    SPECIFICATIONS FOR PARAMETERS
      INTEGER    NOBS
      PARAMETER  (NOBS=25)
C
      INTEGER    NOUT
      REAL       B0, B1, STAT(12), XDATA(NOBS), YDATA(NOBS)
      CHARACTER  CLABEL(13)*15, RLABEL(1)*4
      EXTERNAL   RLINE, UMACH, WRRRL
C
      DATA XDATA/35.3, 29.7, 30.8, 58.8, 61.4, 71.3, 74.4, 76.7, 70.7,
     &     57.5, 46.4, 28.9, 28.1, 39.1, 46.8, 48.5, 59.3, 70.0, 70.0,
     &     74.5, 72.1, 58.1, 44.6, 33.4, 28.6/
      DATA YDATA/10.98, 11.13, 12.51, 8.4, 9.27, 8.73, 6.36, 8.5,
     &     7.82, 9.14, 8.24, 12.19, 11.88, 9.57, 10.94, 9.58, 10.09,
     &     8.11, 6.83, 8.88, 7.68, 8.47, 8.86, 10.36, 11.08/
      DATA RLABEL/'NONE'/, CLABEL/' ', 'Mean of X', 'Mean of Y',
     &           'Variance X', 'Variance Y', 'Corr.',
     &           'Std. Err. B0', 'Std. Err. B1', 'DF Reg.',
     &           'SS Reg.', 'DF Error', 'SS Error', 'Pts. with NaN'/
C
      CALL RLINE (NOBS, XDATA, YDATA, B0, B1, STAT)
C
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) B0, B1
99999 FORMAT (' B0 = ', F7.2, '  B1 = ', F9.5)
      CALL WRRRL ('%/STAT', 1, 12, STAT, 1, 0, '(12W10.4)', RLABEL,
     &            CLABEL)
C
      END
```

Output

```
B0 =   13.62  B1 =   -.07983

                                   STAT
 Mean of X   Mean of Y  Variance X  Variance Y      Corr.  Std. Err. B0
      52.6       9.424       298.1       2.659     -.8452         .5815

Std. Err. B1     DF Reg.     SS Reg.    DF Error   SS Error  Pts. with NaN
      .01052           1       45.59          23      18.22              0
```

Reference

Draper, N. R., and H. Smith (1981), *Applied Regression Analysis*, second edition. John Wiley & Sons, New York.

RLAV/DRLAV (Single/Double precision)

Purpose: Fit a multiple linear regression model using the least absolute values criterion.

Usage:
```
CALL RLAV (NOBS, NCOL, X, LDX, INTCEP, IIND, INDIND,
           IRSP, B, IRANK, SAE, ITER, NRMISS)
```

Arguments

NOBS - Number of observations. (Input)

NCOL - Number of columns in X. (Input)

X - NOBS by NCOL matrix containing the data. (Input)

LDX - Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)

INTCEP - Intercept option. (Input)

INTCEP	Action
0	An intercept is not in the model.
1	An intercept is in the model.

IIND - Independent variable option. (Input)
The absolute value of IIND is the number of independent (explanatory) variables. The sign of IIND specifies the following options:

IIND	Meaning
.LT. 0	The data for the -IIND independent variables are given in the first -IIND columns of X.
.GT. 0	The data for the IIND independent variables are in the columns of X whose column numbers are given by the elements of INDIND.
.EQ. 0	There are no independent variables.

The regressors are the constant regressor (if INTCEP = 1) and the independent variables.

INDIND - Index vector of length IIND containing the column numbers of X that are the independent (explanatory) variables. (Input, if IIND is positive)
If IIND is negative, INDIND is not referenced and can be a vector of length one.

IRSP - Column number IRSP of X contains the data for the response (dependent) variable. (Input)

B - Vector of length INTCEP + IABS(IIND) containing a LAV solution for the regression coefficients. (Output)
If INTCEP = 1, B(1) contains the intercept estimate. B(INTCEP+I) contains the coefficient estimate for the I-th independent variable.

IRANK - Rank of the matrix of regressors. (Output)

```
            If IRANK is less than INTCEP + IABS(IIND), linear
            dependence of the regressors was declared.
SAE     - Sum of the absolute values of the errors.  (Output)
ITER    - Number of iterations performed.  (Output)
NRMISS  - Number of rows of data containing NaN (not a number)
            for the dependent or independent variables.  (Output)
            If a row of data contains NaN for any of these variables,
            that row is excluded from the computations.
```

Remarks

```
1. Automatic workspace usage is
            RLAV     NOBS*(IABS(IIND)+5)+2*IABS(IIND)+NOBS+4, or
            DRLAV    2*NOBS*(IABS(IIND)+5)+4*IABS(IIND)+NOBS+8
   Workspace may be explicitly provided, if desired, by use of
   R2AV/DR2AV.  The reference is
            CALL R2AV (NOBS, NCOL, X, LDX, INTCEP, IIND, INDIND,
                       IRSP, B, IRANK, SAE, ITER, NRMISS, IWK, WK)
   The additional arguments are as follows:
   IWK     - Work vector of length NOBS
   WK      - Work vector of length NOBS*(IABS(IIND)+5)+2*IABS(IIND)+4


2. Informational error
   Type Code
     3   1  The solution may not be unique.


Keywords:   L1 criterion; MSAE; LSAE; MAD; LAV
```

Algorithm

RLAV computes estimates of the regression coefficients in a multiple linear regression model. The criterion satisfied is the minimization of the sum of the absolute values of the deviations of the observed response y_i from the fitted response $\hat{y}_i$ for a set on n observations. Under this criterion, known as the L_1 or LAV (least absolute value) criterion, the regression coefficient estimates minimize $\sum_{i=1}^{n} |y_i - \hat{y}_i|$.

The estimation problem can be posed as a linear programming problem. The special nature of the problem, however, allows for considerable gains in efficiency by the modification of the usual simplex algorithm for linear programming. These modifications are described in detail by Barrodale and Roberts (1973).

In many cases, the algorithm can be made faster by computing a least squares solution prior to the invocation of RLAV. This is particularly useful when a least squares solution has already been computed. The procedure is as follows:

1. Fit the model using least squares and compute the residuals from this fit.

2. Fit the residuals from step 1 on the regressor variables in the model using RLAV.
3. Add the two estimated regression coefficient vectors from steps 1 and 2. The result is an L_1 solution.

When multiple solutions exist for a given problem, subroutine RLAV may yield different estimates of the regression coefficients on different computers; however, the sum of the absolute values of the residuals should be the same (within rounding differences). The informational error indicating nonunique solutions may result from rounding accumulation. Conversely, because of rounding the error may fail to result, even when the problem does have multiple solutions.

Example

A straight line fit to a data set is computed under the LAV criterion.

```
C                                  SPECIFICATIONS FOR PARAMETERS
      INTEGER    LDX, NCOEF, NCOL, NOBS
      PARAMETER  (NCOEF=2, NCOL=2, NOBS=8, LDX=NOBS)
C
      INTEGER    IIND, INDIND(1), INTCEP, IRANK, IRSP, ITER, NOUT,
     &           NRMISS
      REAL       B(NCOEF), SAE, X(LDX,NCOL)
      CHARACTER  CLABEL(1)*4, RLABEL(1)*4
      EXTERNAL   RLAV, UMACH, WRRRL
C
      DATA (X(1,J),J=1,NCOL) /1.0, 1.0/
      DATA (X(2,J),J=1,NCOL) /4.0, 5.0/
      DATA (X(3,J),J=1,NCOL) /2.0, 0.0/
      DATA (X(4,J),J=1,NCOL) /2.0, 2.0/
      DATA (X(5,J),J=1,NCOL) /3.0, 1.5/
      DATA (X(6,J),J=1,NCOL) /3.0, 2.5/
      DATA (X(7,J),J=1,NCOL) /4.0, 2.0/
      DATA (X(8,J),J=1,NCOL) /5.0, 3.0/
C
      INTCEP = 1
      IIND   = -1
      IRSP   = 2
C
      CALL RLAV (NOBS, NCOL, X, LDX, INTCEP, IIND, INDIND, IRSP, B,
     &           IRANK, SAE, ITER, NRMISS)
C
      CALL UMACH (2, NOUT)
      RLABEL(1) = 'B ='
      CLABEL(1) = 'NONE'
      CALL WRRRL (' ', 1, NCOEF, B, 1, 0, '(F6.2)', RLABEL, CLABEL)
      WRITE (NOUT,*) 'IRANK = ', IRANK
      WRITE (NOUT,*) 'SAE = ', SAE
      WRITE (NOUT,*) 'ITER = ', ITER
      WRITE (NOUT,*) 'NRMISS = ', NRMISS
```

```
      END
```

Output

```
B =      .50      .50
IRANK = 2
SAE = 6.
ITER = 2
NRMISS = 0
```

References

Barrodale, I., and F. D. K. Roberts (1973), An improved algorithm for discrete L_1 approximation, *SIAM Journal on Numerical Analysis*, **10**, 839–848.

Barrodale, I., and F. D. K. Roberts (1974), Solution of an overdetermined system of equations in the l_1 norm, *Communications of the ACM*, **17**, 319–320.

DISTRIBUTION LIST

2 Copies to:

Commander
US Army Medical Research and Development Command
SGRD-RMS
Fort Detrick
Frederick, MD 21701

12 Copies to:

Defense Technical Information Center
ATTN: DTIC-DDA
Alexandria, VA 22314

1 Copy to:

Commandant
Academy of Health Sciences, US Army
ATTN: AHS-COM
Fort Sam Houston, TX 78234

1 Copy to:

Dir of Biol & Med Sciences Division
Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217

1 Copy to:

CO, Naval Medical R&D Command
National Naval Medical Center
Bethesda, MD 20014

1 Copy to:

HQ AFMSC/SGPA
Brooks AFB, TX 78235

1 Copy to:

Director of Defense Research and Engineering
ATTN: Assistant Director (Environment and Life Sciences)
Washington, DC 20301

1 Copy to:

Dean
School of Medicine Uniformed Services
University of Health Sciences
4301 Jones Bridge Road
Bethesda, MD 20014

1 Copy to:

HQDA
Training Directorate, O DCSOPS
ATTN: DAMO-TRS/Maj(P) Hayford
WASH, DC 20310-04

1 Copy to:

Director
Army Physical Fitness Research Institute
Box 469
US Army War College
Carlisle Barracks, PA 17013

1 Copy to:

Director
Soldier Physical Fitness School
ATTN: ATSG-PF
US Army Soldier Support Institute
Fort Benjamin Harrison, IN 46216

1 Copy to:

Dr. James Hodgdon
Environmental Physiology Department
US Naval Health Research Center
P.O. Box 85122
San Diego, CA 92138

1 Copy to:

Dr. M. F. Haisman
Head, Applied Physiology
Army Personnel Research Establishment
c/o RAE
Farnborough
Hants, GU14 6TD
United Kingdom